Călin George Ionuț

TIA 2024

Predicția finală (cea mai bună): **0.85810**

Metoda de învățare folosită

Am ales metoda **Random Forest**. Acesta se bazează pe crearea și combinarea mai multor arbori de decizie pentru a îmbunătăți acuratețea și robustețea modelului.

Ne imaginăm o pădure formată din mai mulți arbori. Fiecare arbore reprezintă o opinie independentă despre datele acestea. În final, Random Forest ia decizia pe baza majorității acestor opinii, ceea ce duce la o predicție robustă și precisă.

Astfel, Random Forest combină puterea mai multor modele slabe (arbori de decizie) pentru a crea un model puternic și stabil.

Călin George Ionuț

Codul

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score

class FashionMNISTAnalyzer:
    def __init__(self, train_path, test_path, submission_path):

        train_data = np.load(train_path)
        self.x_train = train_data['x_train']
        self.y_train = train_data['y_train']

        test_data = np.load(test_path)
        self.x_test = test_data["x_test"]

        self.submission = pd.read_csv(submission_path)

    def preprocess_data(self, subset_ratio=0.8, apply_pca=True, n_components=100):

        x_train, _, y_train, _ = train_test_split(
            self.x_train, self.y_train,
            train_size=subset_ratio,
            random_state=42
        )

        x_train, x_val, y_train, y_val = train_test_split(
            x_train, y_train,
            test_size=0.2,
            random_state=42
        )

        scaler = StandardScaler()
        x_train_scaled = scaler.fit_transform(x_train)
        x_val_scaled = scaler.transform(x_val)
        x_test_scaled = scaler.transform(self.x_test)

        if apply_pca:
            pca = PCA(n_components=n_components, random_state=42)
            x_train_scaled = pca.fit_transform(x_train_scaled)
            x_val_scaled = pca.transform(x_val_scaled)
            x_test_scaled = pca.transform(x_test_scaled)

        return (
            x_train_scaled, x_val_scaled,
            y_train, y_val,
            x_test_scaled
        )

    def train_random_forest(self, x_train, y_train):

        rf_classifier = RandomForestClassifier(
            n_estimators=200,
            max_depth=25,
```

```
            max_depth=25,
            min_samples_split=2,
            min_samples_leaf=1,
            bootstrap=True,
            random_state=42,
            n_jobs=-1
        )
        rf_classifier.fit(x_train, y_train)
        return rf_classifier

    def evaluate_model(self, model, x_val, y_val):

        y_pred = model.predict(x_val)
        precision = precision_score(y_val, y_pred, average='macro')
        print(f"Precision: {precision:.4f}")
        return precision

    def generate_predictions(self, model, x_test):

        predictions = model.predict(x_test)
        self.submission['Label'] = predictions
        print("Updated sample_submission.csv with predictions.")
        return self.submission

def main():

    train_path = "train.npz"
    test_path = "test.npz"
    submission_path = "sample_submission.csv"

    analyzer = FashionMNISTAnalyzer(train_path, test_path, submission_path)

    x_train, x_val, y_train, y_val, x_test = analyzer.preprocess_data(
        subset_ratio=0.8,
        apply_pca=True,
        n_components=100
    )

    print("Training Random Forest:")
    rf_model = analyzer.train_random_forest(x_train, y_train)

    print("Evaluating Random Forest Model:")
    precision = analyzer.evaluate_model(rf_model, x_val, y_val)
    print(f"Final Precision Score: {precision:.4f}")

    submission = analyzer.generate_predictions(rf_model, x_test)
    submission.to_csv('final_submission.csv', index=False)

if __name__ == "__main__":
    main()
```

Explicarea Codului

Importarea bibliotecilor necesare:

Clasa FashionMNISTAnalyzer:

- Metoda init: Încarcă datele de antrenament și testare din fișiere .npz și șablonul de trimitere dintr-un fișier .csv
- Metoda preprocess_data: Preprocesează datele prin împărțirea lor în seturi de antrenament și validare, scalarea caracteristicilor și aplicarea PCA pentru reducerea dimensionalității
- Metoda train_random_forest: Antrenează un model Random Forest folosind RandomizedSearchCV pentru a găsi cei mai buni hiperparametri
- Metoda evaluate_model: Evaluează modelul antrenat pe setul de validare și calculează scorul de precizie
- Metoda generate_predictions: Generează predicții pentru setul de testare și actualizează fișierul de trimitere cu aceste predicții

Funcția main:

- Definește căile către fișierele de date
- Inițializează un obiect FashionMNISTAnalyzer
- Preprocesează datele
- Antrenează modelul Random Forest
- Evaluează modelul și afișează scorul de precizie
- Generează predicții pentru setul de testare și salvează rezultatele într-un fișier .csv

Explicația folosirii Random Forest-ului:

Random Forest este un algoritm de învățare automată utilizat pentru clasificare și regresie. Este compus dintr-o colecție de arbori de decizie antrenați pe diferite subseturi ale datelor de antrenament și utilizează medierea pentru a îmbunătăți acuratețea și a controla supraînvățarea (așa cum am explicat și stabilit mai devreme).

Pașii principali în utilizarea Random Forest-ului în acest cod sunt:

- Definirea hiperparametrilor:

  - n_estimators: Numărul de arbori în pădurea aleatorie.
  - max_depth: Adâncimea maximă a fiecărui arbore.
  - min_samples_split: Numărul minim de eșantioane necesare pentru a împărți un nod.
  - min_samples_leaf: Numărul minim de eșantioane necesare pentru a fi la un nod frunză

- Antrenarea modelului:

Se utilizează RandomizedSearchCV pentru a efectua o căutare aleatorie a celor mai buni hiperparametri. Aceasta este o metodă eficientă din punct de vedere al timpului, deoarece nu testează toate combinațiile posibile de hiperparametri, ci doar un subset aleatoriu. Modelul este antrenat pe datele de antrenament preprocesate.

- Evaluarea modelului:

Modelul antrenat este evaluat pe setul de validare pentru a calcula scorul de precizie. Aceasta ajută la determinarea performanței modelului pe date nevăzute.

De ce precizia este 0.85810 și nu mai mare?

- Reducerea dimensiunii cu PCA:

Utilizarea PCA reduce dimensiunea datelor la 100 de componente principale. În timp ce PCA ajută la scăderea timpului de rulare și la eliminarea redundanței, poate duce la

pierderea unor informații relevante, mai ales în cazul imaginilor din Fashion MNIST, unde detalii fine contează.

- Parametrii Random Forest:

Modelul utilizează 200 de arbori și o adâncime maximă de 25. Deși acești parametri sunt solizi, pot să nu fie suficient de puternici pentru a captura toate caracteristicile setului de date.

- Proporția de date folosită pentru antrenament:

Se utilizează doar 80% din date pentru antrenament și 20% pentru validare. Reducerea setului de antrenament poate diminua performanța generală, deoarece Random Forest beneficiază de volume mari de date.

- Scalarea cu StandardScaler:

În unele cazuri, imaginile în format brut (neprocesate prin PCA) pot beneficia mai mult de scalare fără reducerea dimensiunii.

Așadar, precizia este de 0.85810, însă putem modifica programul astfel încât sa obținem o precizie mai bună prin optimizarea parametrilor, sau monitorizarea timpului de rulare, sau creșterea numărului de componente, sau chiar folosirea altei metode de învățare (precum KNN).